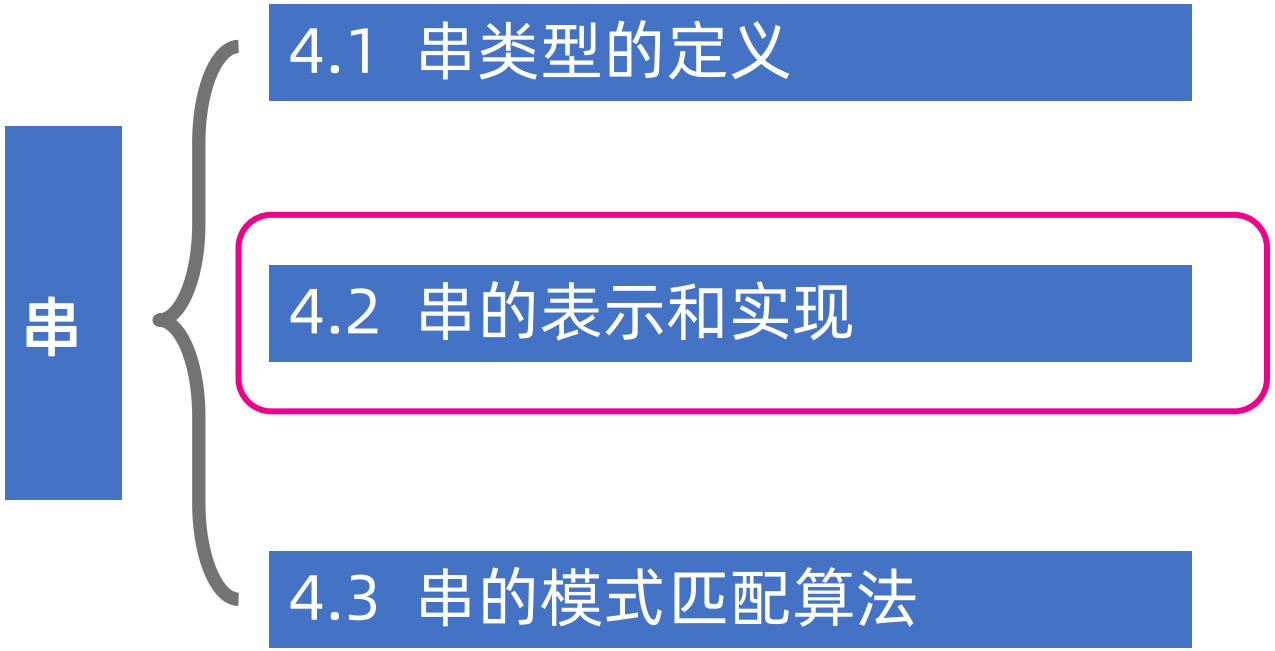
四川師範大學
SICHUAN NORMAL UNIVERSITY

第4章　串

DATA STRUCTURE

计算机科学学院　廖雪花

本章内容简介

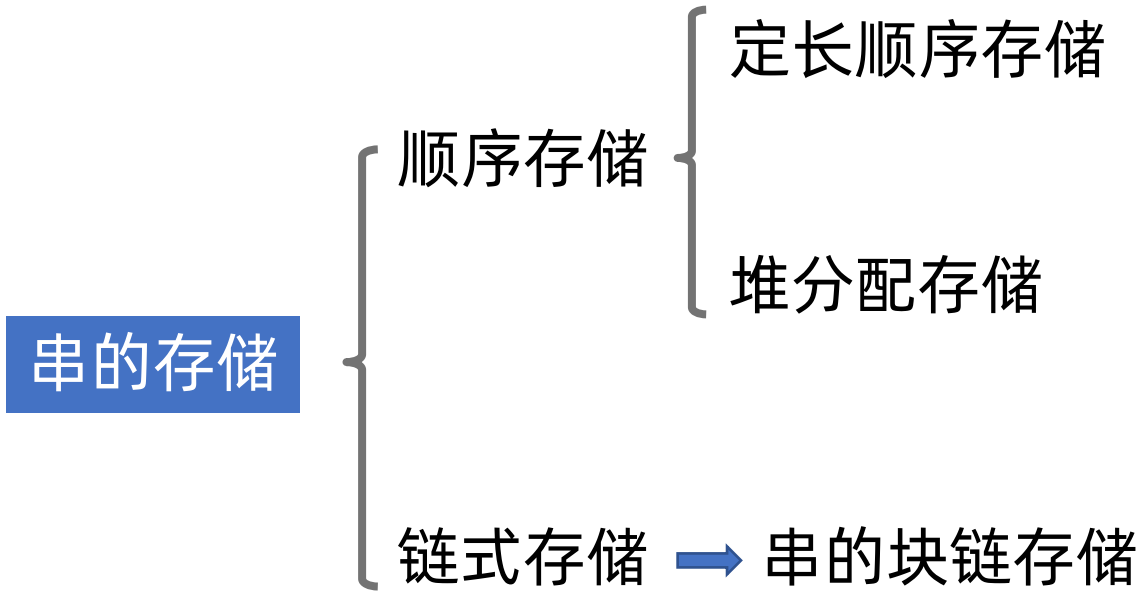


4.2 串的表示和实现

廖雪花　LiaoXuehua

四川師範大學
SICHUAN NORMAL UNIVERSITY

串的3种机内存储表示



4.2.1 定长顺序存储表示

廖雪花　LiaoXuehua

四川師範大學
SICHUAN NORMAL UNIVERSITY

串的定长顺序存储

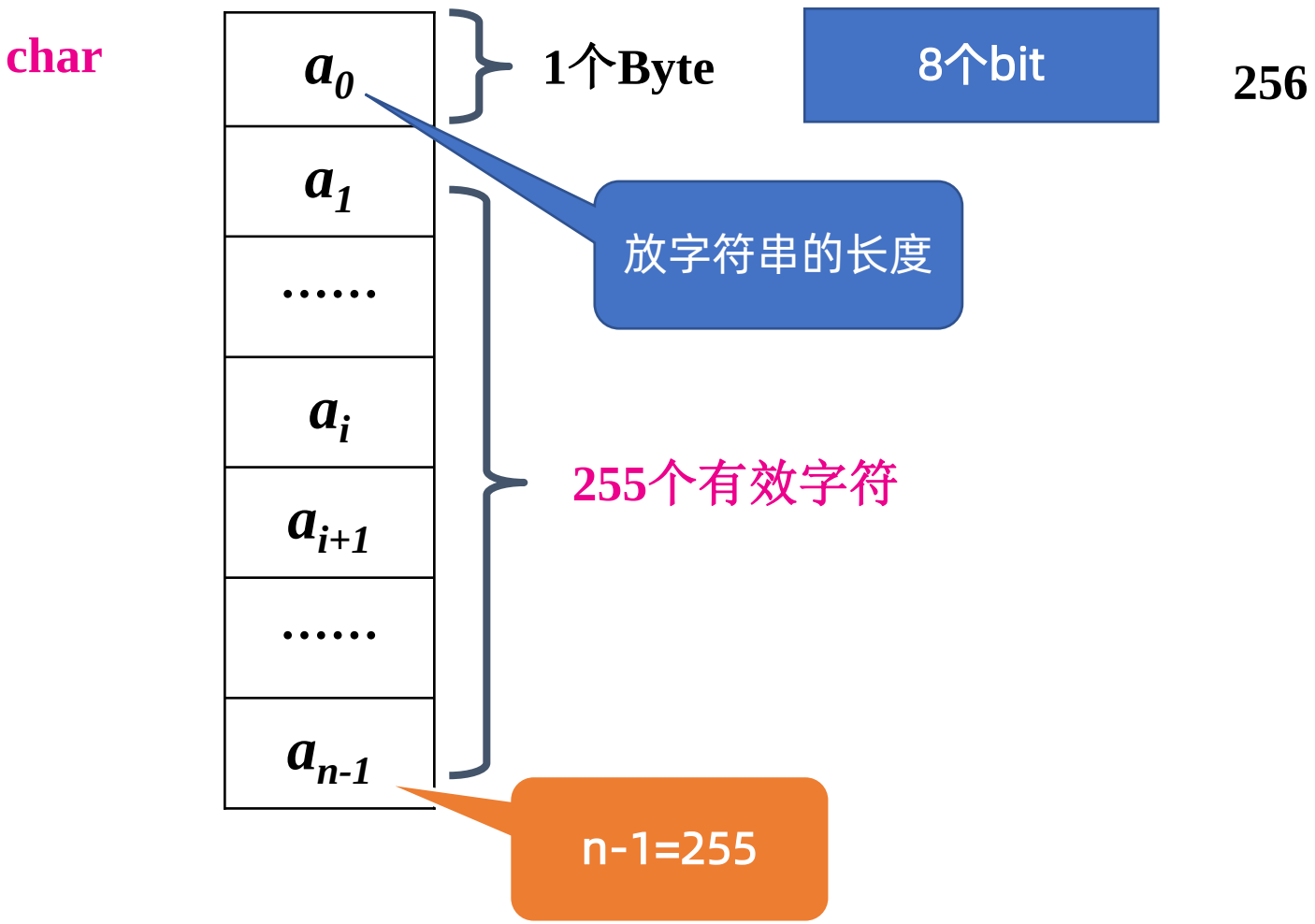
- 定长顺序存储表示,也称为静态存储分配的顺序表。它是用一组连续的存储单元来存放串中的字符序列。

如何表示一个串的结束?

串的定长顺序存储

- 一般可使用一个不会出现在串中的特殊字符在串值的尾部来表示串的结束。
  - 例如，C语言中以字符‘\0’表示串值的终结。
  - 缺点：串长隐含，不便于计算。

串的定长顺序存储


char
a_0
a_1
……
a_i
a_{i+1}
……
a_{n-1}
1个Byte
8个bit
256
放字符串的长度
255个有效字符
n-1=255

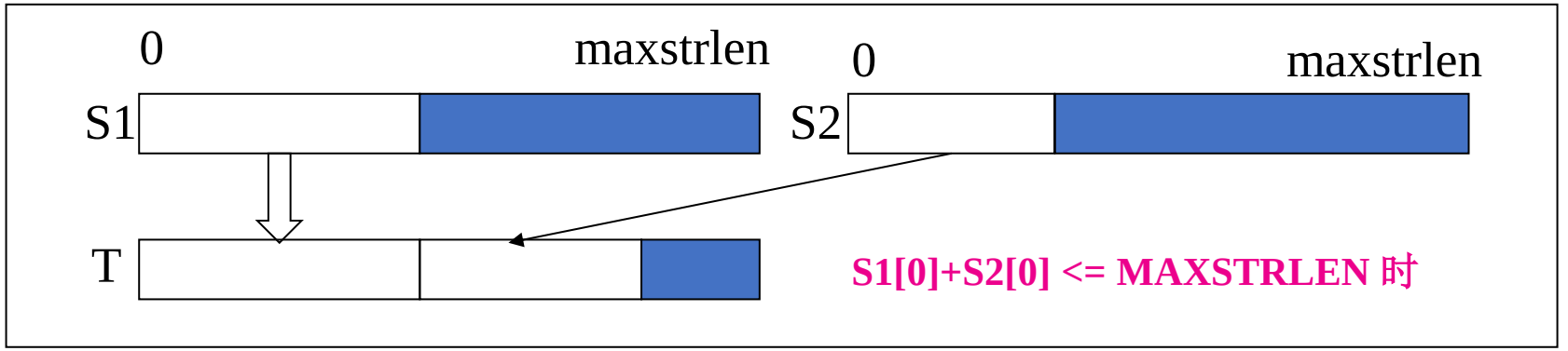
串的定长顺序存储

- 直接使用定长的字符数组来定义，数组的上界预先给出。

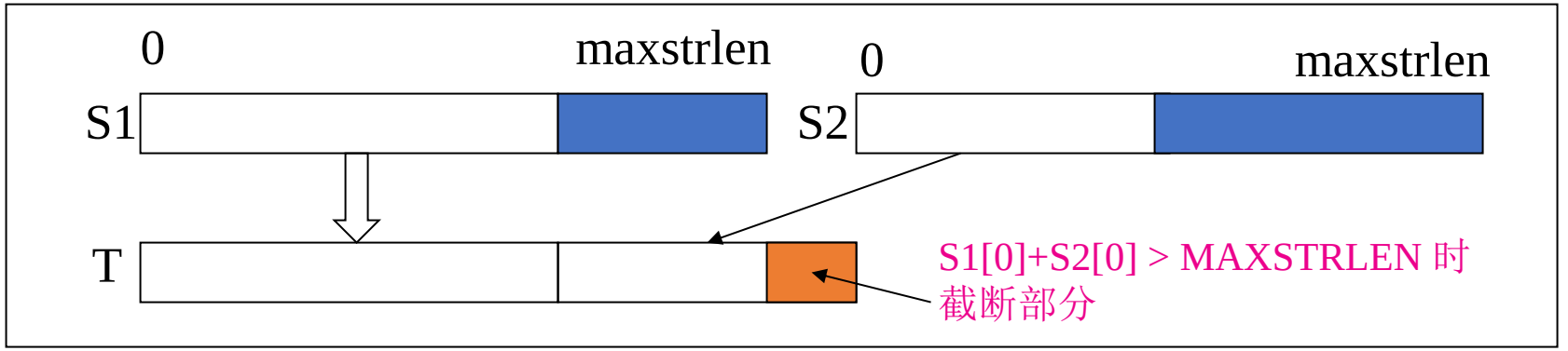
```
#define MAXSTRLEN  255
typedef char SString[MAXSTRLEN +1];
SString s;   //s是一个可容纳255个字符的顺序串。
                  // 0号单元存放串的长度
```

1.串联接： Concat(SString &T, SString S1, SString S2)

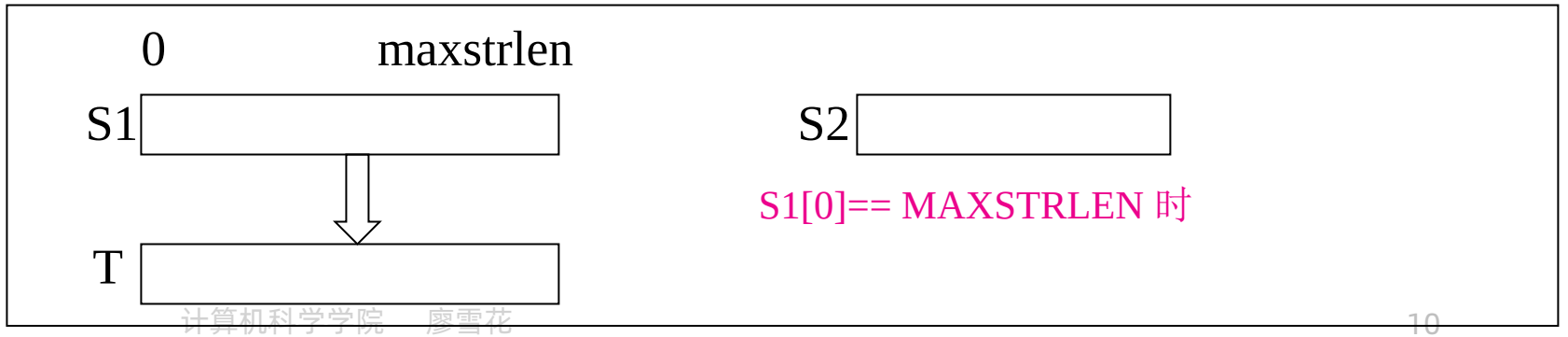
（1）S1[0]+S2[0] <= MAXSTRLEN 时



（2）S1[0]+S2[0] > MAXSTRLEN 时



（3）S1[0]== MAXSTRLEN 时



串联接操作实现

```
Status Concat(SString &T,SString S1,SString S2){
    if (S1[0]+S2[0]<=MAXSTRLEN){   //未截断
        T[1 ...S1[0]]=S1[1 ...S1[0]];
        T[S1[0]+1 ...S1[0]+S2[0]]=S2[1 ...S2[0]];
        T[0]=S1[0]+S2[0];
        uncut=TRUE;}
    else if  (S1[0]<MAXSTRLEN){     //截断，取S2的部分
            T[1 ...S1[0]]=S1[1 ...S1[0]];
            T[S1[0]+1 ...MAXSTRLEN]=S2[1...MAXSTRLEN-S1[0]];
            T[0]= MAXSTRLEN; uncut=FALSE;}
        else {    T[0..MAXSTRLEN]=S1[0...MAXSTRLEN];
                                    //S1[0]=T[0]==MAXSTRLEN
                uncut=FALSE;}  //截断，只取S1
    return OK;
    }
```

2.求子串

```
Status SubString(SString &Sub,SString S,int pos,int len){
    if (pos<1 || pos>S[0] ||len<0||len>S[0]-pos+1)
        return ERROR;
    Sub[1 …len]=S[pos …pos+len-1];
    Sub[0]=len;
    return OK;
}//SubString
```

串的定长顺序存储小结

- 串的定长顺序存储结构。其基本操作为“字符序列的复制”。
- 串的存储结构必须预先定义允许存放串的最大字符个数，容易造成系统空间浪费。
- 串的某些操作（如：串的连接、串的替换等）受到限制(如超长需截尾处理）。

4.2.2 堆分配存储表示

廖雪花　LiaoXuehua

四川師範大學
SICHUAN NORMAL UNIVERSITY

串的堆分配存储

- 堆分配存储表示,也是用一组连续的存储单元存储串值的字符序列。

  但存储空间是在程序执行过程中动态分配得到的。

- 存储表示：

```
typedef struct{
        char *ch;
        int  length;
}HString;
```

注：在C语言中，利用动态存储管理函数，根据实际需要来动态分配和释放字符数组空间。

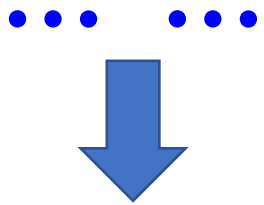
1.串联接： Concat(&T,S1,S2)

```
Status Concat(HString &T, HString S1, HString S2) {
  // 用T返回由S1和S2联接而成的新串
  if (T.ch)  free(T.ch);       // 释放旧空间
  if (!(T.ch = (char *)  malloc((S1.length+S2.length)*sizeof(char))))
      exit (OVERFLOW);
  T.ch[0..S1.length-1] = S1.ch[0..S1.length-1];
  T.length = S1.length + S2.length;
  T.ch[S1.length..T.length-1] = S2.ch[0..S2.length-1];
  return OK;
} // Concat
```

2.求子串：SubString(&Sub,S,pos,len)

```
Status SubString(HString &Sub, HString S, int pos, int len) {
  // 用Sub返回串S的第pos个字符起长度为len的子串
   if (pos < 1  ||  pos > S.length  ||  len < 0  ||  len > S.length-pos+1)
        return ERROR;
   if (Sub.ch)  free (Sub.ch);          // 释放旧空间
   if (!len)
       { Sub.ch = NULL;  Sub.length = 0; } // 空子串
   else {          ...   ...              } // 完整子串
   return OK;
} // SubString
```

2.求子串：SubString(&Sub,S,pos,len)



```
Sub.ch = (char *)malloc(len*sizeof(char));

Sub.ch[0..len-1] = S[pos-1..pos+len-2];

Sub.length = len;
```

本节要点

- 串的堆分配存储
  - 动态分配
  - 顺序存储
- 串连接
- 求子串

4.2.3 串的链式存储结构

廖雪花　LiaoXuehua

四川師範大學
SICHUAN NORMAL UNIVERSITY

串的链式存储

- 顺序串上的插入和删除操作不方便，需要移动大量的字符。因此，我们可用单链表方式来存储串值，串的这种链式存储结构简称为链串。

```
typedef struct node{
        char data;
        struct node *next;
}*LString;
```

串的链式存储

注意：

- 一个链串由头指针唯一确定。
- 这种结构便于进行插入和删除运算，但存储空间利用率太低。

串的块链存储

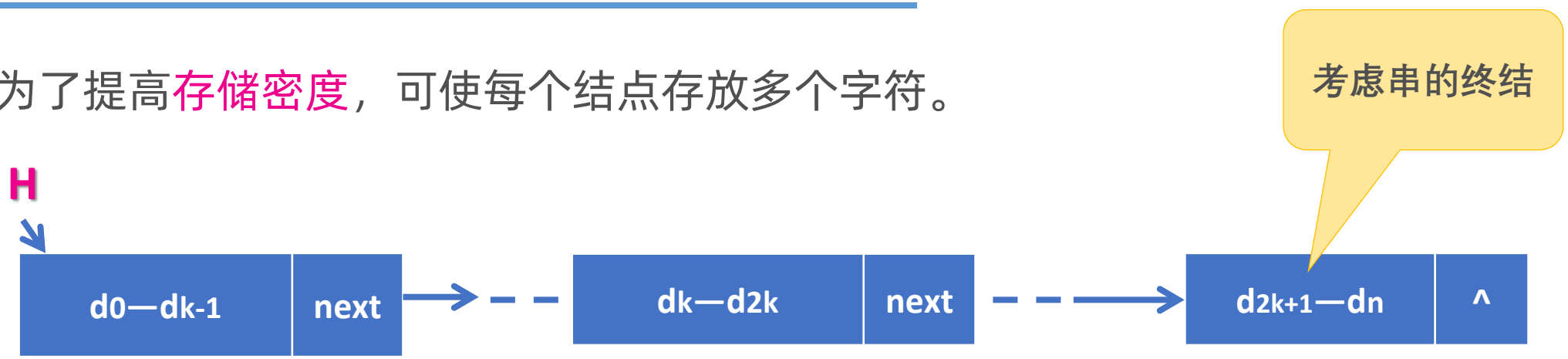
- 为了提高存储空间利用率，可使每个结点存放多个字符。我们用存储密度表示。

- 什么是存储密度？

$$\text{存储密度} = \frac{\text{串值所占的存储位}}{\text{实际分配的存储位}}$$

串的块链存储

- 为了提高存储密度，可使每个结点存放多个字符。



- 通常将结点数据域存放的字符个数定义为结点的大小，显然，当结点大小大于1时，串的长度不一定正好是“结点大小”的整数倍，因此要用特殊字符来填充最后一个结点，以表示串的终结。

串的块链存储

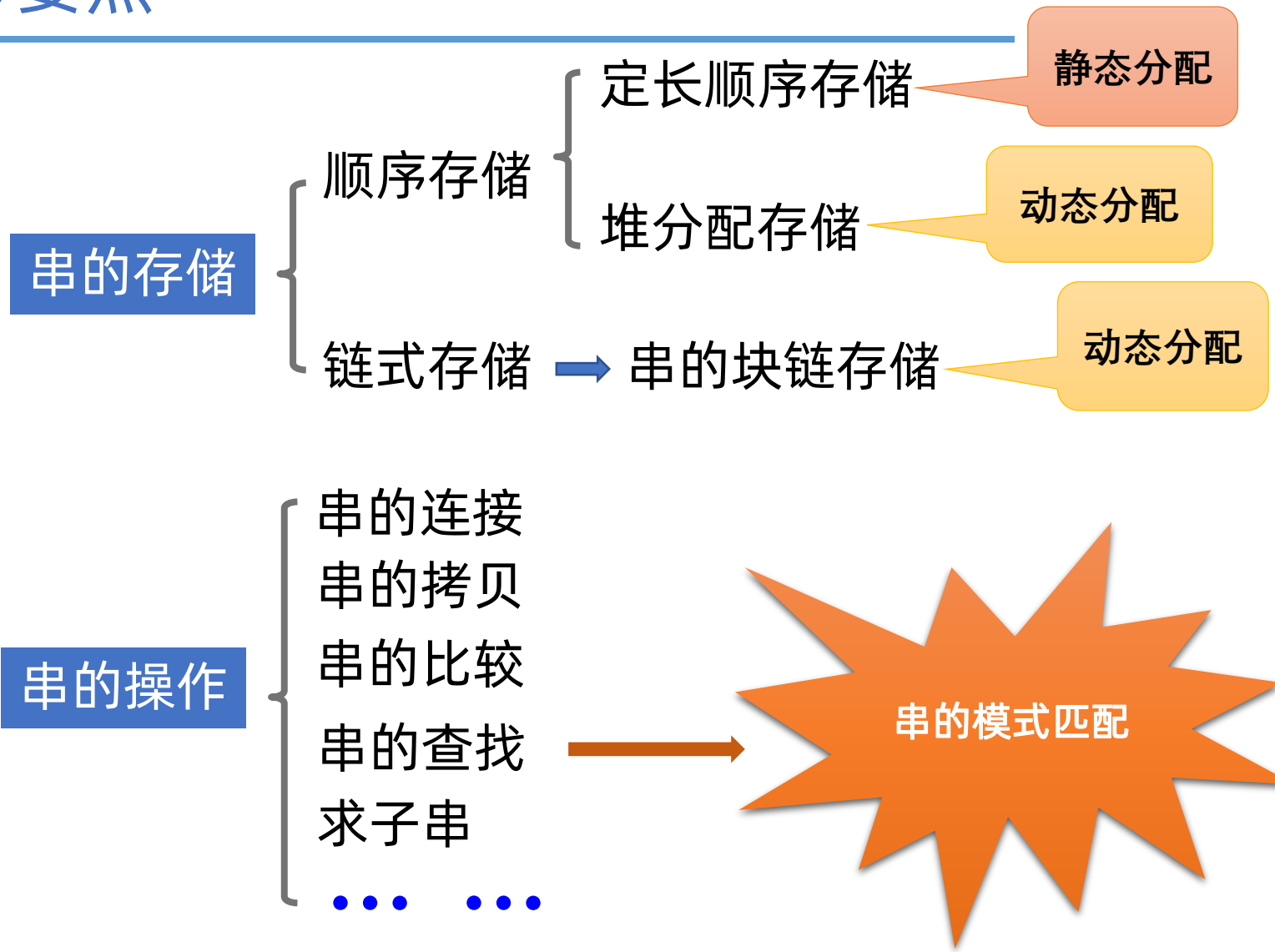
```
#define  CHUNKSIZE  80  // 可由用户定义的块大小
typedef  struct Chunk {  // 结点结构
  char  ch[CUNKSIZE];
  struct Chunk  *next;
} Chunk;
typedef struct {  // 串的链表结构
  Chunk *head, *tail; // 串的头和尾指针
   int   curlen;     // 串的当前长度
} LString;
```

如何确定一个结点的大小?

串的块链存储

- 存储密度小（如结点大小为1时），运算处理方便，然而，存储占用量大。
- 如果要在串的处理过程中进行内外存交换，则会影响处理总效率。
- 串的链式存储结构对某些串操作，如连接等反而不方便。
- 总的说来，串的链式存储不如顺序存储结构灵活，而且占用存储量大、操作复杂。
- 串的链式存储下串操作和链表操作类似，此处不再详细讨论。

本节要点


串的存储
顺序存储
定长顺序存储
静态分配
堆分配存储
动态分配
链式存储
串的块链存储
动态分配
串的操作
串的连接
串的拷贝
串的比较
串的查找
求子串
串的模式匹配

感谢聆听